Save Page Now 2 Public API Docs Draft

Vangelis Banos, updated: 2025-10-22

Capture a web page as it appears now for use as a trusted citation in the future. Changelog: https://docs.google.com/document/d/19RJsRncGUw2qHqGGg9lqYZYf7KKXMDL1Mro5o1Qw6QI/edit#

Contents

Glossary 1

Basic API Reference 1

Capture request 1

Status request 3

Error codes 5

User status 7

System status 7

Tips for faster captures 7

Limitations 8

Example PHP script using the SPN2 API to capture a URL 9

Frequently Asked Questions 10

Glossary

Capture	A record in the Wayback Machine that can be accessed like this: http://web.archive.org/web/20051231203615/http://www.bbc.co.uk/
Timestamp	A datetime format used in the Wayback Machine: YYYYMMDDHHMMSS. Example: 20051231203615
Embeds	Components of a web page, e.g. images, CSS, JS, etc. When we capture a web page, we also try to capture its embeds. We return them with the capture result.
Outlinks	Links found inside the capture. We return them with the capture result.

Basic API Reference

The Save Page Now 2 (SPN2) API enables you to make a **capture request** and then check its progress with a **status request**.

Capture request

SPN2 runs on https://web.archive.org/save which requires authentication using two alternative methods:

1. **S3 API Keys** (highly preferable). Get your account's keys at https://archive.org/account/s3.php Use HTTP Header ***"authorization: LOW myaccesskey:mysecret"*** in your requests.
2. Cookies: Get **logged-in-sig** and **logged-in-user** from your browser when you log in to

https://archive.org and add them to your SPN2 HTTP requests. Cookies are not desirable because they tend to expire after a few days so you would need to login again to archive.org to get new cookies.

To capture a web page via the API, you can use an HTTP POST or GET request as follows:

```
curl -X POST -H "Accept: application/json" -H "Authorization: LOW myaccesskey:mysecret"
-d'url=http://brewster.kahle.org/' https://web.archive.org/save
or
curl -X GET -H "Accept: application/json" --cookie "logged-in-sig=xxx;logged-in-user=user1%40archive.org;"
https://web.archive.org/save/http://brewster.kahle.org/
```

Additional capture request options (HTTP POST required).

Important note: Anything other than "1" or "on" is considered to be "off". If you use "01" or "True" it means "off".

Parameter	Description
capture_all=1	Capture a web page with errors (HTTP status=4xx or 5xx). By default SPN2 captures only status=200 URLs.
capture_outlinks=1	Capture web page outlinks automatically. This also applies to PDF, JSON, RSS and MRSS feeds.
capture_screenshot=1	Capture full page screenshot in PNG format. This is also stored in the Wayback Machine as a different capture.
delay_wb_availability=1	The capture becomes available in the Wayback Machine after ~12 hours instead of immediately. This option helps reduce the load on our systems. All API responses remain exactly the same when using this option.
force_get=1	Force the use of a simple HTTP GET request to capture the target URL. By default SPN2 does a HTTP HEAD on the target URL to decide whether to use a headless browser or a simple HTTP GET request. force_get overrides this behavior.
skip_first_archive=1	Skip checking if a capture is a first if you don't need this information. This will make captures run faster.
if_not_archived_within=<timedelta>	Capture web page only if the latest existing capture at the Archive is older than the <timedelta> limit. Its format could be any datetime expression like "3d 5h 20m" or just a number of seconds, e.g. "120". If there is a capture within the defined timedelta, SPN2 returns that as a recent capture. The default system <timedelta> is 45 min.
if_not_archived_within= <timedelta1>,<timedelta2>	When using 2 comma separated <timedelta> values, the first one applies to the main capture and the second one applies to outlinks.
outlinks_availability=1	Return the timestamp of the last capture for all outlinks.
email_result=1	Send an email report of the captured URLs to the user's email.

js_behavior_timeout=<N>	Run JS code for <N> seconds after page load to trigger target page functionality like image loading on mouse over, scroll down to load more content, etc. The default system <N> is 5 sec. More details on the JS code we execute: https://github.com/internetarchive/brozzler/blob/master/brozzler/behaviors.yaml WARNING: The max <N> value that applies is 30 sec. NOTE: If the target page doesn't have any JS you need to run, you can use js_behavior_timeout=0 to speed up the capture.
capture_cookie=<XXX>	Use extra HTTP Cookie value when capturing the target page.
use_user_agent=<XXX>	Use custom HTTP User-Agent value when capturing the target page.
target_username=<XXX> target_password=<YYY>	Use your own username and password in the target page's login forms.

Example

```
curl -X POST -H "Accept: application/json"
-d'url=http://brewster.kahle.org/&capture_outlinks=1&capture_all=1'  -H "Authorization: LOW
myaccesskey:mysecret" https://web.archive.org/save
```

In any case, a capture request might return:

```
{"url":"http://brewster.kahle.org/", "job_id":"ac58789b-f3ca-48d0-9ea6-1d1225e98695"}
```

Status request

It is possible to see the status of one or multiple captures via the API.
To see a capture status, you can use an HTTP GET or POST request as follows:

```
curl -X GET -H "Accept: application/json" -H "Authorization: LOW myaccesskey:mysecret"
https://web.archive.org/save/status/ac58789b-f3ca-48d0-9ea6-1d1225e98695
or
curl -X POST -H "Accept: application/json" -d'job_id=ac58789b-f3ca-48d0-9ea6-1d1225e98695' --cookie
"logged-in-sig=AAAAAAAAAA;logged-in-user=user1%40archive.org;" https://web.archive.org/save/status
```

In any case, a capture status request might return the following if successful:


```
{"status":"success",
 "job_id":"ac58789b-f3ca-48d0-9ea6-1d1225e98695",
 "original_url":"http://brewster.kahle.org/",
 "screenshot":"http://web.archive.org/screenshot/http://brewster.kahle.org/"
 "timestamp":"20180326070330",
 "duration_sec":6.203,
 "resources":[
   "http://brewster.kahle.org/",
   "http://brewster.kahle.org/favicon.ico",
   "http://brewster.kahle.org/files/2011/07/bkheader-follow.jpg",
   "http://brewster.kahle.org/files/2016/12/amazon-unhappy.jpg",
   "http://brewster.kahle.org/files/2017/01/computer-1294045_960_720-300x300.png",
```

```
    "http://brewster.kahle.org/files/2017/11/20thcenturytimemachineimages_0000.jpg",
    "http://brewster.kahle.org/files/2018/02/IMG_6041-1-300x225.jpg",
    "http://brewster.kahle.org/files/2018/02/IMG_6061-768x1024.jpg",
    "http://brewster.kahle.org/files/2018/02/IMG_6103-300x225.jpg",
    "http://brewster.kahle.org/files/2018/02/IMG_6132-225x300.jpg",
    "http://brewster.kahle.org/files/2018/02/IMG_6138-1-300x225.jpg",
    "http://brewster.kahle.org/wp-content/themes/twentyten/images/wordpress.png",
    "http://brewster.kahle.org/wp-content/themes/twentyten/style.css",
    "http://brewster.kahle.org/wp-includes/js/wp-embed.min.js?ver=4.9.4",
    "http://brewster.kahle.org/wp-includes/js/wp-emoji-release.min.js?ver=4.9.4",
    "http://platform.twitter.com/widgets.js",
    "https://archive-it.org/piwik.js",
    "https://platform.twitter.com/jot.html",
    "https://platform.twitter.com/js/button.556f0ea0e4da4e66cfdc182016dbd6db.js",
    "https://platform.twitter.com/widgets/follow_button.f47a2e0b4471326b6fa0f163bda46011.en.html",
    "https://syndication.twitter.com/settings",
    "https://www.syndikat.org/en/joint_venture/embed/",
    "https://www.syndikat.org/wp-admin/images/w-logo-blue.png",
    "https://www.syndikat.org/wp-content/plugins/user-access-manager/css/uamAdmin.css?ver=1.0",
    "https://www.syndikat.org/wp-content/plugins/user-access-manager/css/uamLoginForm.css?ver=1.0",
    "https://www.syndikat.org/wp-content/plugins/user-access-manager/js/functions.js?ver=4.9.4",
    "https://www.syndikat.org/wp-content/plugins/wysija-newsletters/css/validationEngine.jquery.css?ver=2.8.1",
    "https://www.syndikat.org/wp-content/uploads/2017/11/s_miete_fr-200x116.png",
    "https://www.syndikat.org/wp-includes/js/jquery/jquery-migrate.min.js?ver=1.4.1",
    "https://www.syndikat.org/wp-includes/js/jquery/jquery.js?ver=1.12.4",
    "https://www.syndikat.org/wp-includes/js/wp-emoji-release.min.js?ver=4.9.4"
  ],
  "outlinks":{
    "https://archive.org/": "xxxxxx89b-f3ca-48d0-9ea6-1d1225e98695",
    "https://other.com": "yyyy89b-f3ca-48d0-9ea6-1d1225e98695"
  }}
```


Note that *"original_url":"http://brewster.kahle.org/"* contains the final URL **after following potential redirects**.

Note that *"screenshot":"http://web.archive.org/screenshot/http://brewster.kahle.org/"* is included in the response only when we use **capture_screenshot=1**. In case there is a screenshot capture error, the result doesn't include a "screenshot" field.

When **outlinks_availability=1** option is used, the outlinks would be like the following:


```
"outlinks":{
    "https://archive.org/": {"timestamp": "20180102005040"},
    "https://other.com": {"timestamp": "20190102005040"},
    "https://other-not-captured.com": {"timestamp": null}
  }
```


In case the capture is pending, it may return:


```
{"status":"pending",
  "job_id":"e70f33c7-9eca-4c88-826d-26930564d7c8",
  "resources":[
    "https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js",
    "https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.21/jquery-ui.min.js",
    "https://cdn.onesignal.com/sdks/OneSignalSDK.js",
  ]
}
```

In case there is an error, it may return:


```
{"status":"error",
  "exception":"[Errno -2] Name or service not known",
  "status_ext":"error:invalid-host-resolution",
  "job_id":"2546c79b-ec70-4bec-b78b-1941c42a6374",
  "message":"Couldn't resolve host for http://example5123.com.",
  "resources": []
}
```


Error codes

The error codes and messages may vary depending on the problem. Field **status_ext** contains more information on the specific error type.

status_ext	Description
error:bad-gateway	Bad Gateway for URL (HTTP status=502).
error:bad-request	The server could not understand the request due to invalid syntax. (HTTP status=401)
error:bandwidth-limit-exceeded	The target server has exceeded the bandwidth specified by the server administrator. (HTTP status=509).
error:blocked	The target site is blocking us (HTTP status=999).
error:blocked-client-ip	Anonymous clients which are listed in https://www.spamhaus.org/xbl/ or https://www.spamhaus.org/sbl/ lists (spams & exploits) are blocked. Tor exit nodes are excluded from this filter.
error:blocked-url	We use a URL block list based on Mozilla web tracker lists to avoid unwanted captures.
error:browsing-timeout	SPN2 back-end headless browser timeout.
error:capture-location-error	SPN2 back-end cannot find the created capture location. (system error).
error:cannot-fetch	Cannot fetch the target URL due to system overload.
error:celery	Cannot start capture task.
error:filesize-limit	Cannot capture web resources over 2GB.
error:ftp-access-denied	Tried to capture an FTP resource but access was denied.
error:gateway-timeout	The target server didn't respond in time. (HTTP status=504).
error:http-version-not-supported	The target server does not support the HTTP protocol version used in the request for URL (HTTP status=505).
error:internal-server-error	SPN internal server error.
error:invalid-url-syntax	Target URL syntax is not valid.

error:invalid-server-response	The target server response was invalid. (e.g. invalid headers, invalid content encoding, etc).
error:invalid-host-resolution	Couldn’t resolve the target host.
error:job-failed	Capture failed due to system error.
error:method-not-allowed	The request method is known by the server but has been disabled and cannot be used (HTTP status=405).
error:not-implemented	The request method is not supported by the server and cannot be handled (HTTP status=501).
error:no-browsers-available	SPN2 back-end headless browser cannot run.
error:network-authentication-required	The client needs to authenticate to gain network access to the URL (HTTP status=511).
error:no-access	Target URL could not be accessed (status=403).
error:not-found	Target URL not found (status=404).
error:not-implemented	The request method is not supported by the server and cannot be handled for URL (HTTP status=501).
error:proxy-error	SPN2 back-end proxy error.
error:protocol-error	HTTP connection broken. (A possible cause of this error is "IncompleteRead").
error:read-timeout	HTTP connection read timeout.
error:soft-time-limit-exceeded	Capture duration exceeded 45s time limit and was terminated.
error:service-unavailable	Service unavailable for URL (HTTP status=503).
error:too-many-daily-captures	This URL has been captured 10 times today. We cannot make any more captures.
error:too-many-redirects	Too many redirects. SPN2 tries to follow 3 redirects automatically.
error:too-many-requests	The target host has received too many requests from SPN and it is blocking it. (HTTP status=429). Note that captures to the same host will be delayed for 10-20s after receiving this response to remedy the situation.
error:user-session-limit	User has reached the limit of concurrent active capture sessions.
error:unauthorized	The server requires authentication (HTTP status=401).
error:max-daily-bandwidth	An authenticated user can archive up to 5GB per day.
error:max-daily-bandwidth-from-ip	An anonymous user can archive up to 2GB per day.
error:max-daily-bandwidth-host	SPN2 can archive up to 100GB per day from a host.

In case you used option `capture_outlinks=1`, the result outlinks include the job_id for each outlink so that you could check its status later. Else, outlinks key contains the list of URLs only.

You can access the created capture using the following URL pattern:

```
https://web.archive.org/web/<timestamp>/<original_url>
```

Advanced status request usage
To see the status of **multiple captures**, use parameter **job_ids** and a comma separated list of values:

```
curl -X POST -H "Accept: application/json"
-d'job_ids=ac58789b-f3ca-48d0-9ea6-1d1225e98695,ac58789b-f3ca-48d0-9ea6-xxxxxx,
ac58789b-f3ca-48d0-9ea6-yyyyyyyyy' --cookie
"logged-in-sig=AAAAAAAAAA;logged-in-user=user1%40archive.org;" https://web.archive.org/save/status
```

To see the capture status of all outlinks, use parameter **job_id_outlinks** and the job_id of the parent capture:

```
curl -X POST -H "Accept: application/json" -d'job_id_outlinks=ac58789b-f3ca-48d0-9ea6-1d1225e98695'
--cookie "logged-in-sig=AAAAAAAAAA;logged-in-user=user1%40archive.org;"
https://web.archive.org/save/status
```

User status

You can see the current number of active and available session of your user account using the following:

```
curl -X GET -H "Accept: application/json" -H "Authorization: LOW myaccesskey:mysecret"
http://web.archive.org/save/status/user
```

To avoid getting a stale cache response, it is better to use a URL like this:
http://web.archive.org/save/status/user?_t=1602606392499 where _t is a random variable.

The response will be like:

```
{"available":12,"processing":3}
```

System status

You can check if the service is overloaded using the following:

```
curl -X GET -H "Accept: application/json" http://web.archive.org/save/status/system
```

If everything is fine, it may return:

```
{"status":"ok"}
```

If the service is overloaded, it may return:

```
{"status":"Save Page Now servers are temporarily overloaded. Your captures may be delayed."}
```

To be clear, SPN will still work fine in this case, besides some delays.

If there is a critical problem, there will be an HTTP status=502 response.

Tips for faster captures

The following options have a real impact on the speed of your captures.

- If you don't need to know if your capture is the first in the Archive, please use **skip_first_archive=1.**
- If you are sure that the target URL is not an HTML page and can be downloaded via a plain HTTP request, use option **force_get=1.**
- If the target HTML page is plain and you don't need to run any JS behavior to download all content (JS behaviors scroll down the page automatically and/or trigger AJAX requests), use **js_behavior_timeout=0.**
- Do NOT use **capture_outlinks=1** unless it is really necessary to capture all outlinks. If you are interested in capturing a specific outlink, make a capture, check the list of outlinks returned by SPN2 and capture only the specific outlink(s) you need.

Limitations

The operation of SPN2 is limited in several ways as described in the following table. The aim of these limitations is to ensure the performance and stability of the application.

Limitation	**Description**
Network connection timeout = 10s	If we try connecting to a target URL and it takes more than 10s to respond, we consider the server unresponsive and return a capture error.
Max captures per minute for authenticated users = 12 and for anonymous users = 4.	Any user can do N captures per minute. If you try to start more, SPN2 returns an error.
Max web page capture time = 50s	SPN2 browsers can spend up to 50s visiting a target URL and running JS behaviors. If web page capture hasn't finished after that time, we terminate the browser and check if we have downloaded sufficient content to consider this a successful capture.
Max capture duration = 2m	The total time spent capturing any URL cannot be over 2m.
Max JS behavior runtime = 7s (configurable)	The total time running JS events (scroll down, mouse over, etc) cannot be over 5s by default. This is configurable using param: *js_behavior_timeout=<N>*
Max redirects = 3	SPN2 tries to follow redirects automatically.
Max resource size = 2GB	The max file size SPN2 can download.
Max number of outlinks captured using capture_outlinks option = 100	SPN2 captures the first N outlinks automatically when using option capture_outlinks. Outlinks are ordered using some rules before selecting the first N: 1. PDF

	2. Epub 3. URLs containing substrings "new" or "update" 4. URLs of the same domain as the original capture URL. Please note that if you don't use option capture_outlinks, you get a list of all outlinks without any filtering or ranking. You could use that list to download any URLs necessary.
Max number of outlinks returned = 1000	SPN2 just returns a list of outlinks if "capture outlinks" is not selected. This list is limited to 1000 items.
Max number of embeds returned = 1000	SPN2 tracks all captured embeds and lists them in "resources". This list is limited to 1000 items.
Max number of links captured from emails in spn@archive.org = 500	SPN2 tries to capture the first 500 links in emails sent to spn@archive.org.
Max captures per day for anonymous users = 4k	Anonymous users can use SPN2 but their total captures per day cannot be more than this limit.
Max captures per day for authenticated users = 100k	The captures of authenticated users cannot be more than this limit per day. If you need to make more captures, please contact info@archive.org.
Max captures per day for a URL = 10	It is possible to capture the same URL only 10 times per day.
Blocked URLs	SPN2 uses Mozilla web tracker block lists to avoid capturing some URLs. You may get an "error:blocked-url" when trying to make a capture.
Artificial delays for multiple concurrent captures on the same host.	When we run more than 20 concurrent captures on the same host, we introduce an artificial delay on subsequent captures to avoid overloading the target and blocking SPN2. The delay algorithm is: When concurrent_capture_number > 20 for the same host, delay concurrent_capture_number/5 sec. For example: if concurrent_capture_number = 50, delay a new capture by 50/5 = 10 sec.
Max emails processed by spn@archive.org service per user per day= 10	You can send HTML emails with links to capture at spn@archive.org. The system processes 10 emails per user per day and discards the rest.
Max screenshot size is 4MB	If you select "Save screen shot" and its size is > 4MB, it is skipped to avoid system overload.
Max captures' size per day is 2GB for anonymous users.	Anonymous users can capture no more than 2GB total per day.
Max captures' size per day is 5GB for authenticated users.	Authenticated users can capture no more than 5GB total per day.

Example PHP script using the SPN2 API to capture a URL

```
<?php
/**
 * Example PHP script which captures a URL via the SPN2 API.
 * Note that this script doesn't include proper exception handling and is not
 * optimised for production use.
 * Tested with PHP 7.0 and the PHP curl extension on Ubuntu 16.04.
 *
 * Full SPN2 API reference:
 * https://docs.google.com/document/d/1Nsv52MvSjbLb2PCpHlat0gkzw0EvtSgpKHu4mk0MnrA/edit
 *
 * Archive.org credentials are required to use the SPN2 API,
 * get your credentials from https://archive.org/account/s3.php
 */
$KEY = "XXX";
$SECRET = "YYY";
$TARGET_URL = "https://bbc.co.uk";

$headers = array("Accept: application/json",
                 "Content-Type: application/x-www-form-urlencoded;charset=UTF-8",
                 "Authorization: LOW {$KEY}:{$SECRET}");
$params = array('url'=>$TARGET_URL);

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, "https://web.archive.org/save");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($params));
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($ch);
curl_close($ch);
$data = json_decode($response, true);
$job_id = $data['job_id'];
print("Capture started, job id: {$job_id}\n");
while(true) {
    sleep(5);
    $response = file_get_contents("http://web.archive.org/save/status/{$job_id}");
    $data = json_decode($response, true);
    if ($data['status'] == 'success') {
        print("Capture complete: https://web.archive.org/web/{$data['timestamp']}/{$data['original_url']}\n");
        break;
    } else if ($data['status'] == 'error') {
        print("Error: {$data['message']}\n");
        break;
    }
    print("Wait, still capturing...\n");
}
```

Frequently Asked Questions

Q1. I can see the page http://example.com/ from my web browser but when I try to capture it, I get an error: "Live page is not available".

Before SPN2 captures a URL, it tries to do a quick HTTP HEAD and if that fails an HTTP GET to see if the target URL is online. If these requests fail, we return an error: *"Live page is not available"*. If they are successful, we make the capture using our headless browser. Also, we cache the result for 10 min to speedup this check for subsequent requests. This check may return an invalid result for many reasons:

1. The site may have blocked requests from IA IPs in general.
2. We are doing many captures on the same site at the same time (e.g. by other SPN2 users or via "capture outlinks"), the target site receives too many connections from SPN2 and its firewall/web server blocks them. In these cases, the capture result would be "Live page is not available" but the site would be perfectly fine for you as you are using it from your home IP address. To mitigate this issue, we are delaying captures from the same host when there are 50+ concurrent captures.
3. Sites may actually be down for a few seconds or more due to technical issues on their end (e.g. network outages, server problems, etc).

Q2. I'm trying to capture a web page that contains a lot of links using the "capture outlinks" option but no outlinks are captured.

SPN2 can extract outlinks from many file types: HTML pages, PDF, RSS, XML and JSON files. For each file type, it runs a special link extractor software for 30 sec. For HTML pages, it's a JS script that extracts URLs from `a[href], area[href], a[onclick], a[ondblclick]`:
https://github.com/internetarchive/brozzler/blob/master/brozzler/js-templates/extract-outlinks.js
If SPN2 cannot extract outlinks from a URL, one of the following issues may occur:

1. The outlink extraction couldn't finish processing in 30 sec and was terminated.
2. The total URL capture took too long (the limit is 90 sec) and there wasn't time to run the outlink extraction in time.
3. The target URL doesn't have links or they are encoded in a way that is not supported by the outlink extraction software (e.g. using some obscure HTML element attributes and events or an encrypted PDF).

Q3. When I try to do a capture, I get a message saying *"Your capture will begin in XXs."*. Is SPN2 overloaded?

When we run more than 20 concurrent captures on the same host, we introduce an artificial delay on subsequent captures to avoid overloading the target and blocking SPN2. The delay algorithm is:

When concurrent_capture_number > 20 for the same host, delay concurrent_capture_number/5 sec.
For example: if concurrent_capture_number = 50, delay a new capture by 50/5 = 10 sec.

By "concurrent captures", we mean captures performed in the last 60 sec.

In addition to that, if a target site returns HTTP status=429 (too many requests), we delay any subsequent captures for 10 to 20 sec. This rule applies for 60 sec after receiving the status=429 response.